浙江大学

计算机视觉(本科)作业报告

作业名称:	**制作无声小短片视频**
姓 名:	曹小川
学 号:	3200105705
电子邮箱:	434446127@qq.com
联系电话:	18991060018
导 师:	潘纲

2022年 11 月 21 日

Harris Corner Detection

一、 作业已实现的功能简述及运行简要说明

读入摄像头或者一段视频，并讲当前帧图像做一次Harris Corner检测，并将检测结果叠加到原来的图像上。另开一个回放视频窗口，播放视频的最小特征值图。

二、 作业的开发与运行环境

说明/列出开发与运行环境信息，例如开发集成环境、操作系统、各种开发工具SDK、或数据库系统等的名称及版本号。

开发工具:Visual Studio

操作系统: Windows 10

编程语言：C++/ OpenCV

三、 系统或算法的基本思路、原理、及流程或步骤等

首先创建一个`videoCapture`来进行摄像头或者视频的读入，如果摄像头可以使用，那么读入摄像头，否则读入一段视频。接下来对于视频的每一帧进行处理，使用Harris算法来边缘检测。算法首先对帧进行灰度处理，然后使用Sobel算子进行梯度处理，然后对处理后的图像计算X,Y,XY方向的梯度的平方，然后对其进行高斯模糊，使用Harris算法进行计算，最后对其非极大抑制处理。

四、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等

该lab所采用的核心算法是首先对帧进行灰度处理，然后使用Sobel算子进行梯度处理，然后对处理后的图像计算X,Y,XY方向的梯度的平方，然后对其进行高斯模糊，使用Harris算法进行计算，最后对其非极大抑制处理。

```
灰度处理：直接调用cvtColor函数
```

`cvtColor(srcImage, srcGray, COLOR_BGR2GRAY);`

Sobel梯度处理：

`Sobel(srcGray, imageSobelX, CV_8UC1, 1, 0, 3);`

然后将其转化为无符号8字节储存

`convertScaleAbs(imageSobelX, imageSobelX);`

接着计算Sobel梯度在X,Y,XY方向的平方

```
Sobel2(imageSobelX, imageSobelXX);
Sobel2(imageSobelY, imageSobelYY);
SobelXY(imageSobelX, imageSobelY, imageSobelXY);
```

```
void Sobel2(const Mat src, Mat_<float>& dst) {
    dst = Mat(src.size(), CV_32FC1);
    for (int i = 0; i < dst.rows; i++) {
        for (int j = 0; j < dst.cols; j++) {
            dst.at<float>(i, j) = src.at<uchar>(i, j) * src.at<uchar>(i, j);
        }
    }
}
```

然后调用高斯模糊函数对帧进行高斯模糊

```
GaussianBlur(imageSobelXX, GaussianXX, { 3,3 }, 1);
GaussianBlur(imageSobelYY, GaussianYY, { 3,3 }, 1);
GaussianBlur(imageSobelXY, GaussianXY, { 3,3 }, 1);
```

使用Harris算法提取角点

$$Tr(M) = a + b = A + B$$

$$Det(M) = a * b = A * B - C^2$$

```
void harris_G(Mat_<float>& GaussXX, Mat_<float>& GaussYY, Mat_<float>& GaussXY,
Mat_<float>& res, float k) {
    res = Mat_<float>(GaussXX.size(), CV_32FC1);
    for (int i = 0; i < res.rows; i++) {
        for (int j = 0; j < res.cols; j++) {
            float x = GaussXX.at<float>(i, j);
            float y = GaussYY.at<float>(i, j);
            float z = GaussXY.at<float>(i, j);
            res.at<float>(i, j) = x * y - z * z - k * (x + y) * (x + y);
        }
    }
}
```

最后使用非极大值抑制对图像进行处理

```
vector<Point> LocalMaxValue(Mat_<float>& res, Mat& src, int kSize) {
    int l = kSize / 2;
    res = src.clone();
    vector <Point> v;
    for (int i = l; i < res.rows - l; i++) {
        for (int j = l; j < res.cols - l; j++) {
            if (res.at<float>(i, j) > res.at<float>(i - 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j + 1) &&
                res.at<float>(i, j) > res.at<float>(i, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i, j + 1) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j + 1)) {
                if ((int)res.at<float>(i, j) > 1800000) {
```

```
                    v.push_back ( Point(j, i));
                    //cout << "1" << endl;
                }
            }
        }
    }
    return v;
}
```

五、 实验结果与分析

程序代码：

```
#include<opencv2/opencv.hpp>
#include<iostream>
#include <stdlib.h>
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <ctype.h>
using namespace cv;
using namespace std;

vector<Point> op_Frame(Mat srcImage);
void get_Sobel(Mat& imageSource, Mat& imageSobelX, Mat& imageSobelY);
void Sobel2(const Mat imageGradX, Mat_<float>& SobelAmpXX);
void SobelXY(const Mat imageGradX, const Mat imageGradY, Mat_<float>&
SobelAmpXY);
void harris_G(Mat_<float>& GaussXX, Mat_<float>& GaussYY, Mat_<float>& GaussXY,
Mat_<float>& resultData, float k);
vector<Point> LocalMaxValue(Mat_<float>& resultData, Mat& srcGray, int kSize);
void print_Point(vector<Point> v, Mat& ResultImage);
Mat grayToRGB(const Mat input_img);
//Mat res;
int main()
{
    VideoCapture cap(0);
    if (!cap.isOpened()) {
        cout << "Can't load the Cam" << endl;
        cap.open("arc.mp4");
    }
    if (!cap.isOpened()) {
        cout << "Can't load the video" << endl;
        return -1;
    }
    int flag = 0;
    vector<Point> v;
    Mat frame;
    Mat photo;
    while (cap.isOpened()) {

        bool success = cap.read(frame);
        if (success == true) {
```

```
            int key = waitKey(24);
            if (key == 'p') {
                flag = 1;
                v = op_Frame(frame);
                frame.copyTo(photo);
                print_Point(v, photo);
            }
            imshow("Frame", frame);
            if (flag != 0) {
                imshow("Photo", photo);
            }
        }
        else {
            cout << "disconnected" << endl;
            break;
        }
        int key = waitKey(24);
        if (key == 'q') {
            break;
        }
    }

    cap.release();
    destroyAllWindows();
    return 0;
}
vector<Point> op_Frame(Mat srcImage) {

    Mat srcGray;
    cvtColor(srcImage, srcGray, COLOR_BGR2GRAY);
    //ConvertRGB2GRAY(srcImage, srcGray);
    Mat imageSobelX;// = Mat::zeros(srcGray.size(), CV_32SC1);
    Mat imageSobelY;
    Mat resultImage;
    Mat_<float> imageSobelXX;
    Mat_<float> imageSobelYY;
    Mat_<float> imageSobelXY;
    Mat_<float> GaussianXX;
    Mat_<float> GaussianYY;
    Mat_<float> GaussianXY;
    Mat_<float> HarrisRespond;
    Mat_<float> res;
    Mat_<float> R_graph;
    //计算Soble的XY梯度
    Sobel(srcGray, imageSobelX, CV_8UC1, 1, 0, 3);
    convertScaleAbs(imageSobelX, imageSobelX);
    Sobel(srcGray, imageSobelY, CV_8UC1, 0, 1, 3);
    convertScaleAbs(imageSobelY, imageSobelY);
    Sobel2(imageSobelX, imageSobelXX);
    Sobel2(imageSobelY, imageSobelYY);
    SobelXY(imageSobelX, imageSobelY, imageSobelXY);
    GaussianBlur(imageSobelXX, GaussianXX, { 3,3 }, 1);
    GaussianBlur(imageSobelYY, GaussianYY, { 3,3 }, 1);
    GaussianBlur(imageSobelXY, GaussianXY, { 3,3 }, 1);
```

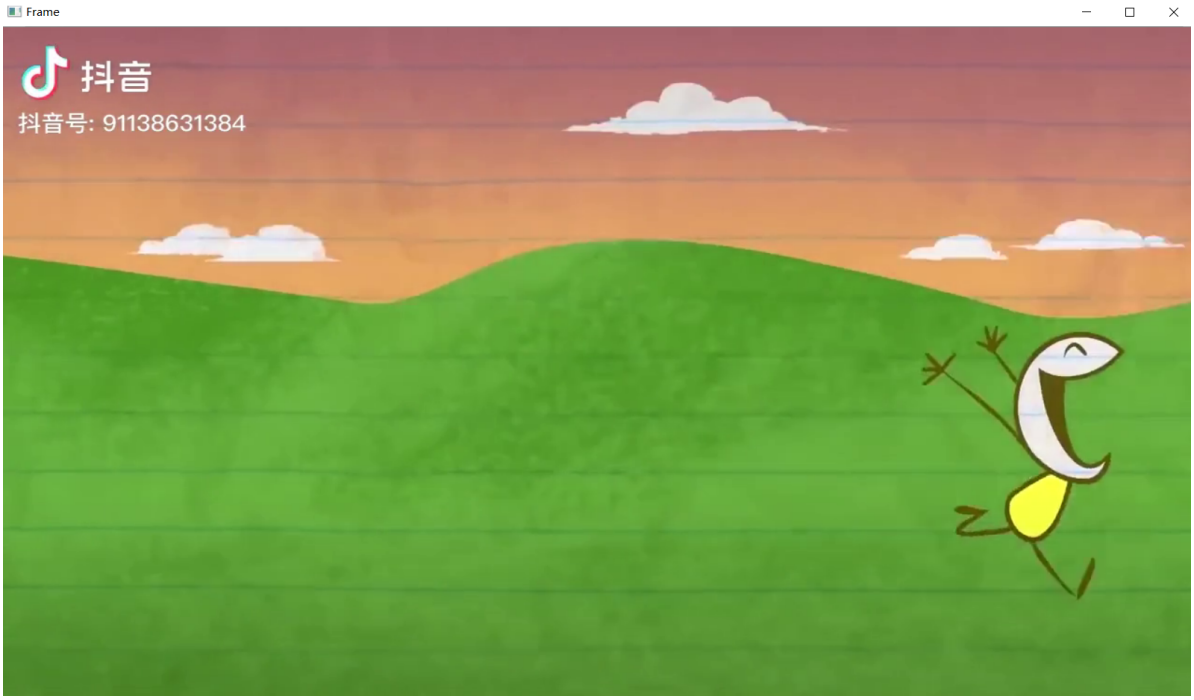
```
    harris_G(GaussianXX, GaussianYY, GaussianXY, HarrisRespond, 0.05);
    resultImage = srcGray.clone();
    vector<Point> v_P = LocalMaxValue(res, HarrisRespond, 3);

    return v_P;
}
void get_Sobel(Mat& imageSource, Mat& imageSobelX, Mat& imageSobelY) {
    imageSobelX = Mat::zeros(imageSource.size(), CV_32SC1);
    imageSobelY = Mat::zeros(imageSource.size(), CV_32SC1);
}
void Sobel2(const Mat src, Mat_<float>& dst) {
    dst = Mat(src.size(), CV_32FC1);
    for (int i = 0; i < dst.rows; i++) {
        for (int j = 0; j < dst.cols; j++) {
            //src.at<int>(i, j);
            dst.at<float>(i, j) = src.at<uchar>(i, j) * src.at<uchar>(i, j);
        }
    }
}
void SobelXY(const Mat src1, const Mat src2, Mat_<float>& dst) {
    dst = Mat(src1.size(), CV_32FC1);
    for (int i = 0; i < dst.rows; i++) {
        for (int j = 0; j < dst.cols; j++) {
            dst.at<float>(i, j) = src1.at<uchar>(i, j)*src2.at<uchar>(i,j);
        }
    }
}
void harris_G(Mat_<float>& GaussXX, Mat_<float>& GaussYY, Mat_<float>& GaussXY,
Mat_<float>& res, float k) {
    res = Mat_<float>(GaussXX.size(), CV_32FC1);
    for (int i = 0; i < res.rows; i++) {
        for (int j = 0; j < res.cols; j++) {
            float x = GaussXX.at<float>(i, j);
            float y = GaussYY.at<float>(i, j);
            float z = GaussXY.at<float>(i, j);
            res.at<float>(i, j) = x * y - z * z - k * (x + y) * (x + y);
        }
    }
}
vector<Point> LocalMaxValue(Mat_<float>& res, Mat& src, int kSize) {
    int l = kSize / 2;
    res = src.clone();
    vector <Point> v;
    for (int i = l; i < res.rows - l; i++) {
        for (int j = l; j < res.cols - l; j++) {
            if (res.at<float>(i, j) > res.at<float>(i - 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i - 1, j + 1) &&
                res.at<float>(i, j) > res.at<float>(i, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i, j + 1) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j - 1) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j) &&
                res.at<float>(i, j) > res.at<float>(i + 1, j + 1)) {
                if ((int)res.at<float>(i, j) > 18000) {
```

```
                    v.push_back ( Point(j, i));
                }
            }
        }
    }
    return v;
}
void print_Point(vector<Point> v , Mat& ResultImage) {
    for (int i = 0; i < v.size(); i++) {
        circle(ResultImage, Point(v[i].x, v[i].y), 2, Scalar(0, 0, 0), 2, 8, 0);
    }
}
Mat grayToRGB(const Mat input_img)
{
    cv::Mat three_channel = cv::Mat::zeros(input_img.rows, input_img.cols,
CV_8UC3);
    std::vector<cv::Mat> channels;
    for (int i = 0; i < 3; i++)
        channels.push_back(input_img);
    cv::merge(channels, three_channel);
    return three_channel;
}
```

效果截图：

输入一个视频：



按p间对其进行角点检测

六、 结论与心得体会

本次实验使用Harris 算法进行角点检测，因为以前没有接触过相关编程，所以我还是花费了很多时间进行对算法以及openCV相关接口的了解，在实验的初期我没有对Sobel进行平方处理，以及非极大值抑制的处理，导致识别的图像效果较差。在通过查阅资料以后对其应用了这两种操作，识别效果能够好一些，但是有一些边缘识别较为薄弱，同时因为视频导入后每一帧都要进行识别操作，所以最终视频显得有些卡顿，所以最后我选择对用户希望检测的单独的帧进行检测。通过本次实验，我对于Harris算法有了更深入的理解，同时更加熟练地掌握了openCV，收获颇丰。

七、 参考文献

OpenCV教程--VideoCapture

Harris角点检测原理详解